Trabalho 2 - Detecção e reconhecimento de placas de trânsito em vias públicas

Johannes Peter Schulte
johpetsc@gmail.com
150132662

Departamento de Ciência da Comptutação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil,

Abstract

O projeto visa aplicar conceitos de visão computacional em *machine learning* para treinar modelos capazes de detectar placas de trânsito e reconhecer suas classes do ponto de vista de um veículo trafegando em vias públicas. A base de dados utilizada contém milhares de imagens em ambientes e condições variadas que, por meio dos métodos aplicados, fez com que o modelo consiga predizer as classes treinadas com acurácia próxima de 99.99%.

1 Introdução

Avanços tecnológicos impõem mudanças significativas na sociedade e no modo de viver das pessoas, e a medida que o mundo se transforma e novos conhecimentos são adquiridos, novas demandas e expectativas são direcionadas à ciência e à indústria. Exemplo disso é o setor automotivo, que anteriormente priorizava desenvolver carros mais rápidos e potentes, mas agora se adapta à uma sociedade mais consciente para o bem estar humano e do planeta.

Além dos carros elétricos, outro grande passo para o futuro dos automóveis são os veículos autônomos, que serão um grande avanço para a sociedade não apenas em conforto e conveniência, mas também pela segurança, reorganização do espaço urbano e melhoria da mobilidade urbana. [1, 5] Veículos autônomos são conduzidos por computadores e por isso há o potencial de se moverem com maior precisão, consciência dos seus arredores e seguindo regras de trânsito. Com isso, existiriam consideravelmente menos acidentes de trânsito (uma das maiores causas de mortes no mundo) [6], os veículos transitariam com maior eficiência, ocupando menos espaço (algumas cidades perdem até 60% do seu espaço para acomodar veículos) [12] e promovendo uma melhor mobilidade urbana (em sua grande maioria, atrasos no trânsito são causados por erros humanos). [1, 5, 12]

Para alcançar a automação, um dos maiores desafios é criar sistemas confiáveis, que sejam capazes de entender o mundo ao seu redor como, por exemplo, a sinalização de trânsito. Com essa finalidade, é necessário que os veículos sejam capazes de detectar e reconhecer a sinalização presente nos locais onde trafegam, para que possam tomar decisões de acordo.

Na visão computacional, métodos de *deep learning* são utilizados em imagens para alcançar tais objetivos, especialmente as redes neurais convolucionais (CNN). [8]

CNN é uma classe de redes neurais que se utiliza de uma rede de neurônios inspirado na forma como o córtex visual dos animais funciona, se mostrando muito eficiente e com ótimos resultados para problemas como os que serão abordados nesse projeto. [8] Modelos para detecção, como R-CNN, Fast R-CNN e Faster R-CNN, possuem diferentes arquiteturas para extrair propostas de regiões em imagens, que então passam pela rede CNN para definir as características da região e fazer sua classificação. [7]

2 Metodologia

O projeto tem como objetivo principal o treinamento de um modelo que seja capaz de reconhecer a classe correta de uma placa de trânsito utilizando a biblioteca *fastai* que fornece ferramentas para *deep learning* em *Python*. Todos os resultados foram produzidos em um computador com CPU i7-4770k, 16GB de RAM e GPU NVIDIA GTX 1660 6GB. Por mais que não seja o ponto central, um modelo de reconhecimento de placas não possui a mesma utilidade por conta própria do que teria em conjunto com detecção, e por isso foi utilizado em conjunto um modelo pré-treinado para detecção de placas, junto com algumas técnicas implementadas para melhorias.

2.1 Reconhecimento

2.1.1 Dados

Placas de trânsito vistas de um veículo se apresentam em várias condições que dificultam seu reconhecimento, como diferentes distâncias (baixa resolução de imagem), iluminação, vandalismo ou até mesmo alguns obstáculos, como folhas de árvores, que podem comprometer a visibilidade. Tendo isso em mente, é preciso utilizar uma base de dados com várias imagens e que possa abranger essas diferentes condições durante o treinamento.

Figure 1: Exemplos de imagens da base de dados GTSRB

A base de dados utilizada se chama "*The German Traffic Sign Recognition Benchmark*", ou GTSRB, que foi utilizada para competições de redes neurais em 2011. Ela conta com 43 diferentes classes de placas encontradas na Alemanha e cerca de 50 mil imagens no total. Por sua grande quantidade de imagens, é possível chegar a resultados muito bons, porém também conta com alguns defeitos que serão discutidos posteriormente. [9]

2.1.2 Modelo

As imagens para o modelo foram carregadas e preparadas com ferramentas da *fastai*, de forma que cada classe foi dividida entre imagens e um *label*, que, neste caso, é o nome da pasta onde se encontram as imagens de cada classe. As pastas foram renomeadas da base

original para que os resultados possam ser apresentados com maior clareza. Além disso, as imagens foram divididas de forma aleatória em imagens de treinamento e imagens de validação numa proporção de 4 imagens de treino para cada imagem de validação, resultando em, aproximadamente, 40 mil para treino e 10 mil para validação e, por último, redimensionadas para o treinamento.

A rede utilizada para o aprendizado foi a CNN (*Convolutional Neural Network*), que se apresenta como um dos melhores classes de redes neurais atualmente para processos de aprendizado que utilizam imagens digitais. As CNNs são muito eficientes para a classificação de imagens devido a sua arquitetura de pesos compartilhados e sua característica de invariante por translação. [4, 8]

Uma CNN emprega a operação matemática conhecida por convolução, no lugar de multiplicação de matrizes e funciona por meio de perceptrons multicamadas que são regularizadas para reduzir o problema de *overfitting* durante o treinamento, já que cada neurônio de uma camada se conecta com todos os outros da próxima camada. O formato de entrada *tensor*, com uma dimensão para cada imagem e suas dimensões, e que depois de passar pela camada de convolução é abstraído para um *feature map* com características para reconhecimento. [4, 8]

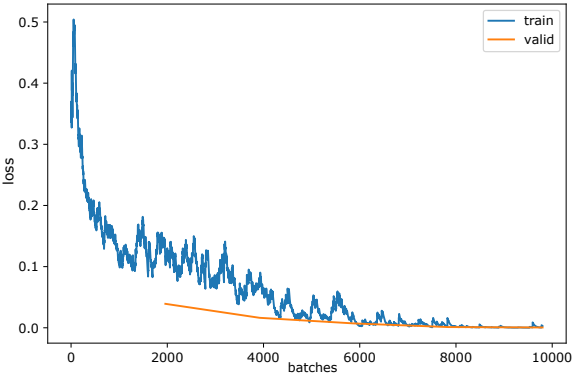

Figure 2: Taxa de perda durante o treinamento com ResNet50

O treinamento utilizando a função de aprendizado em CNN da *fastai* foi feito utilizando os modelos ResNet, redes que utilizam os pulos entre conexões, ou camadas, para acelerar o processamento durante os estágios do treinamento. As diferentes arquiteturas se referem ao número de camadas, onde o maior número pode resultar em treinamentos mais eficientes, pelo custo de um processamento maior, como: ResNet-18, ResNet-34, ResNet-50, etc. [11]

Para encontrar a forma mais eficientes de treinar o modelo, considerando o resultado e o custo de tempo e processamento, foram alterados alguns elementos: o tamanho de cada *batch* para treinamento, a quantidade de iterações do treinamento (*fine tuning*) e a quantidade de camadas do modelo ResNet, de forma que a taxa de perda (Figura 2) para treinamento e validação ao final do treinamento sejam o mais próximas possíveis, significando um treinamento com pouco *overfitting* ou *underfitting*.

Os resultados podem variar por alguns outros fatores, os principais sendo pela inicialização aleatória dos valores, e o embaralhamento dos dados de treinamento implementado pelo algoritmo da *fastai*. Entretanto, foi possível perceber alguns padrões nos diferentes treinamentos realizados, onde a alteração dos valores melhorava consideravelmente o resultado até o ponto de estagnação.

2.2 Detecção

2.2.1 Dados

Para a detecção de placas, a mesma universidade alemã que disponibiliza a base de dados GTSRB, também disponibiliza uma base de dados com o foco em detecção de placas "*The German Traffic Sign Detection Benchmark*", ou GTSDB, onde são disponibilizadas 900 imagens do ponto de vista de um veículo com placas demarcadas para o treinamento. [3]

Figure 3: Exemplos de imagens da base de dados GTSDB

Com a complexidade envolvida no treinamento de modelos para detecção e o foco do projeto voltado para o reconhecimento, foi utilizado um modelo pré-treinado disponibilizado para uso na página *Github* de um artigo para modelos de detecção usando a base de dados GTSDB dividida em três classes para detecção. [13]

2.2.2 Modelo

O modelo utilizado foi treinado dividindo as 43 classes da base de dados de reconhecimento em apenas três, que são agrupadas de acordo com seu formato e cores: uma classe para placas triangulares com contorno vermelho; uma classe para placas circulares com contorno vermelho; uma classe para placas circulares e azuis. [13] Para modelo de extração de características, o treinamento utilizou a rede *Faster R-CNN* (modelo *faster rcnn inception resnet v2 atrous* da biblioteca TensorFlow).

Figure 4: Três classes de detecção

Faster R-CNN é uma evolução dos modelos *R-CNN* e *Fast R-CNN*, algoritmos para gerar propostas de regiões, que são então utilizadas para classificação com CNN. Como a divisão de uma imagem em várias regiões diferentes exige muito processamento, o modelo utilizado implementa a rede RPN, que reduz significativamente o tempo para gerar as regiões em imagens. [7]

Como o modelo foi treinado em três classes de características bem definidas, a detecção entre as classes é bem precisa, porém também apresenta falhas para diferenciar objetos que não são placas mas possuem as mesmas características de cor e forma. Outo problema é que placas com padrão diferentes não são detectadas como, por exemplo, a placa de preferência (um triângulo de cabeça para baixo) e de pare (um octógono).

Para amenizar o problema de algumas placas não serem detectadas, algumas adaptações foram feitas ao código, como a detecção da placa preferência (*yield*), que é uma das classes com mais imagens para treino, e é muito mais utilizada na sinalização alemã que a placa de

pare. [9, 10] Para isso, o modelo de detecção processa a imagem duas vezes, sem rotação e com rotação de 180°, fazendo com que as placas de preferência também sejam detectadas, por apresentarem as mesmas características da primeira classe de detecção apresentada na Figura 4.

3 Resultados

O modelo de reconhecimento foi treinado várias vezes alterando o tamanho de *batch*, a quantidade de iterações e a quantidade de camadas ResNet observando a acurácia ao final do treino, a quantidade de processamento utilizado e o tempo para realizar o treinamento. Foi observado que *batches* grandes utilizavam mais memória do computador, demoravam menos mas também perdiam muita acurácia ao final do treinamento. O maior valor de *batch* testado foi 128, com uma perda de aproximadamente 2.6%, enquanto o melhor resultado foi 16, onde valores menores não apresentavam melhor acurácia e os tempos para treinar o modelo aumentavam sem nenhum ganho significativo.

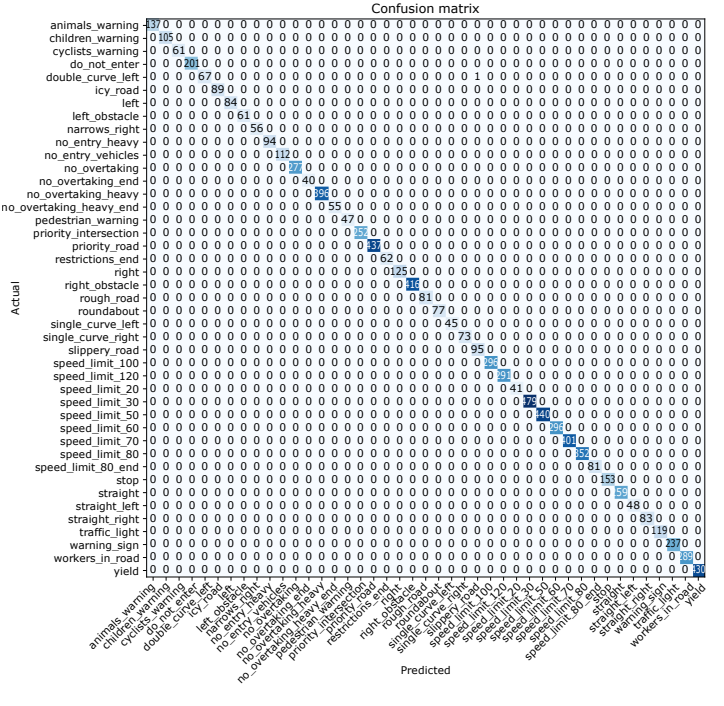

Figure 5: Matriz de confusão do treinamento com ResNet-50

Para a quantidade de iterações utilizadas no *fine tuning*, foi observado que até 4 iterações a taxa de perda do treino diminui consideravelmente, do 4 para 5 apresenta um incremento considerável para alguns casos, 6 o modelo entra em estagnação e valores maiores começam a apresentar maior perda devido ao *overfitting*. É possível observar o comportamento do treinamento durante cada iteração na Figura 2, onde cada iteração equivale a 1960 *batches* para treino e 491 para validação.

Com o tamanho de *batch* de 16 e 5 iterações, foram treinados modelos para diferentes

valores ResNet, onde o único salto significativo foi da ResNet-18 para ResNet-34, que apresentava resultados as vezes tão bom quantos a ResNet-50, porém com uma menor frequência. ResNet-101 demorava mais, não apresentou melhores resultados e tinha maior tendência de *overfitting*. Depois da ResNet-50 o processamento e tempo aumentaram sem nenhuma real melhoria, e por isso foi o modelo escolhido, com o melhor resultado próximo de 99.99%, errando apenas uma predição do conjunto de validação, como pode ser visto na matriz de confusão da Figura 5.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.519074	0.225177	0.927815	0.072185	04:40
0	0.113239	0.039164	0.986736	0.013264	04:53
1	0.075906	0.016292	0.995409	0.004591	05:08
2	0.016052	0.007121	0.998215	0.001785	05:08
3	0.011468	0.001163	0.999490	0.000510	05:06
4	0.002867	0.000483	0.999872	0.000128	05:23

Table 1: Resultado do treinamento com 16 *batches*, 5 iterações e ResNet-50

A Figura 6 mostra as predições com maior perda durante o treinamento. Nela é possível visualizar o bom resultado do modelo treinado, apresentando apenas um erro e quatro predições abaixo de 90%. Também é observado que o modelo é capaz de reconhecer placas até em situações não ideais, como placas com visão obstruída, pouca iluminação e mau tempo. Porém, também é observável as maiores dificuldades do modelo: reconhecer desenhos no centro das placas com baixa resolução e os números dentro de uma placa de velocidade, já que essas apresentam as mesmas características com pequenas diferenças entre os números.

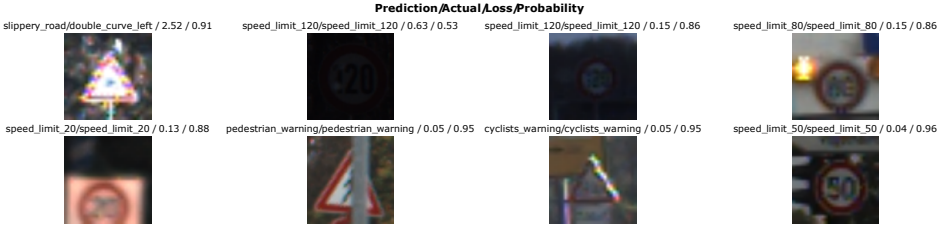

Figure 6: Maiores perdas observadas no treinamento com ResNet-50

O modelo de detecção apresenta uma grande precisão para detectar as classes de placas treinadas, mas é possível observar nos casos de falha que apresenta um grande viés para formas e cores. Esse viés era esperado pela forma como o treinamento foi feito, onde as classes foram divididas entre placas com a mesma forma e cor, isso faz com que o treinamento dê um maior peso para essas características e ignore o conteúdo no centro da placa.

Esse problema em princípio não devia atrapalhar o resultado final, já que o modelo de reconhecimento foi treinado justamente para suprir essa falta de características que diferenciam as placas. Porém, os momentos de falha da detecção também expõem um problema do modelo de reconhecimento, que não foi treinado para reconhecer objetos que não são placas. Isso faz com que, nos casos onde um objeto diferente de uma placa é detectado, o modelo de reconhecimento tenta predizer a placa que o objeto se parece mais.

4 Conclusão

O produto final do projeto, quando o modelo de detecção e reconhecimento são utilizados em conjunto, apresentou resultados muito bons levando em consideração o que os modelos foram treinados para fazer. O modelo de detecção tem uma ótima capacidade de detectar as classes de placas treinadas, e quando o reconhecimento é aplicado, os resultados estão próximos de 100% com bastante frequência, mesmo em condições não ideais.

Figure 7: Exemplo de detecção e reconhecimento de placas

As limitações que o projeto apresenta, mesmo que em uma frequência baixa, mostram que ainda existem vários pontos que podem ser melhorados. Para diminuir os problemas de detecção, o modelo poderia ser treinado com uma classe de placas que não segue o mesmo padrão das utilizadas no treinamento. Outro ponto interessante, seria um modelo treinado para reconhecer objetos que não são placas, de modo a diferenciar as características não apenas de uma placa em relação a outras, mas as de placas em geral em relação a objetos parecidos (mesma cor ou mesma forma).

Para o reconhecimento, um dos problemas do modelo é que a base de dados não é recente, então é possível perceber que algumas placas, principalmente em cidades grandes, já apresentam um visual atualizado. Além de modernizar a base de dados, também seria interessante acrescentar outras classes de placas, já que, no contexto do projeto, a Alemanha possui mais de 300 placas de trânsito oficiais, comparadas as 43 presentes na base de dados. As placas disponíveis para o treinamento são consideravelmente mais presentes nas ruas, fazendo com que essa deficiência não seja tão perceptível.

Outro grande desafio para alcançar o cenário ideal, onde a detecção e reconhecimento são feitos em tempo real, seria diminuir a quantidade de processamento necessário para cada imagem, ou *frame* de vídeo. Isso pode ser feito majoritariamente com melhorias na detecção, como diminuir a quantidade de regiões considerando a probabilidade de placas em cada região da imagem e utilizando modelos mais eficientes. Mesmo com suas limitações, os resultados ainda foram muito bons, e demonstram o potencial que os métodos aplicados no projeto possuem nesse contexto.

References

[1] Adriano Alessandrini, Andrea Campagna, Paolo Delle Site, Francesco Filippi, and Luca Persia. Automated vehicles and the rethinking of mobility and cities. *Transportation Research Procedia*, 5:145 – 160, 2015. ISSN 2352-1465. doi: https://doi.org/10.1016/j.trpro.2015.01.002. URL http://www.sciencedirect.com/science/article/pii/S2352146515000034. SIDT Scientific Seminar 2013.

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[3] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, 2013.

[4] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1):1–207, 2018.

[5] Jonas Meyer, Henrik Becker, Patrick M. Bösch, and Kay W. Axhausen. Autonomous vehicles: The next jump in accessibilities? *Research in Transportation Economics*, 62:80 – 91, 2017. ISSN 0739-8859. doi: https://doi.org/10.1016/j.retrec.2017.03.005. URL http://www.sciencedirect.com/science/article/pii/S0739885917300021.

[6] World Health Organization. The top 10 causes of death, 2018. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[8] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[9] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[10] Route to Germany. Road signs in Germany, 2019. https://routetogermany.com/drivingingermany/road-signs.

[11] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

[12] Roman Zakharenko. Self-driving cars will change cities. *Regional Science and Urban Economics*, 61:26 – 37, 2016. ISSN 0166-0462. doi: https://doi.org/10.1016/j.regsciurbeco.2016.09.003. URL http://www.sciencedirect.com/science/article/pii/S016604621630182X.

[13] Álvaro Arcos-García, Juan A. Álvarez García, and Luis M. Soria-Morillo. Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing*, 316:332 – 344, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.08.009. URL http://www.sciencedirect.com/science/article/pii/S092523121830924X.